

ОГЛАВЛЕНИЕ

ФОРМУЛИРОВКА ЗАДАНИЙ.....	3
ВЫПОЛНЕНИЕ ЗАДАНИЙ.....	4
Часть 1.....	4
Часть 2.....	9
ВЫВОД.....	12

ФОРМУЛИРОВКА ЗАДАНИЙ

Задания на выполнение **1 части** практической работы:

1. Напишите сценарий, который выводит дату, время, список зарегистрировавшихся пользователей, и uptime системы и сохраняет эту информацию в файл.
2. Напишите сценарий, который выводит содержимое любого каталога или сообщение о том, что его не существует.
3. Напишите сценарий, который с помощью цикла прочитает файл и выведет его содержимое.
4. Напишите сценарий, который с помощью цикла выведет список файлов и директорий из текущего каталога, укажет, что есть файл, а что директория.
5. Напишите сценарий, который подсчитает объем диска, занимаемого директорией. В качестве директории можно выбрать любую директорию в системе.
6. Напишите сценарий, который выведет список всех исполняемых файлов в директории, для которых у текущего пользователя есть права на исполнение.

Задания на выполнение **2 части** практической работы:

1. Определение зависимостей проекта.
2. Создание виртуального окружения.
3. Написание скрипта запуска приложения на новой системе.

ВЫПОЛНЕНИЕ ЗАДАНИЙ

Часть 1

Напишем сценарий, который выводит дату, время список зарегистрировавшихся пользователей, и uptime системы и сохраняет эту информацию в файл.

```
prac_2 > $ script1.sh
1  #!/bin/bash
2  LOGFILE="loger.txt"
3
4  {
5      echo "Дата и время: $(date)"
6      echo "Список зарегистрированных пользователей:"
7      ls -l ~/... | grep '^d' | awk '{print $NF}'
8      echo "Время работы системы:"
9      awk '{print $1, "seconds"}' /proc/uptime
10
11 }> "$LOGFILE"
12
13 cat "$LOGFILE"
14
15 echo "Информация сохранена в $LOGFILE"
16
17
```

Рисунок 1 – Исходный код

```

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
$ ./prac_2/script1.sh
Дата и время: Wed, Mar 19, 2025  5:05:42 PM
Список зарегистрированных пользователей:
Default
Public
semen
Время работы системы:
13516.96 seconds
Информация сохранена в logger.txt

```

Рисунок 2 – Пример выполнения

```

≡ logger.txt
1  Дата и время: Wed, Mar 19, 2025  5:05:42 PM
2  Список зарегистрированных пользователей:
3  Default
4  Public
5  semen
6  Время работы системы:
7  13516.96 seconds
8

```

Рисунок 3 – Содержание файла с результатом программы

Напишем сценарий, который выводит содержимое любого каталога или сообщение о том, что его не существует.

```

prac_2 > $ script2.sh
1  #!/bin/bash
2
3  ls "$1"
4

```

Рисунок 4 – Исходный код

```

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
❌ $ ./prac_2/script2.sh
ls: cannot access '': No such file or directory

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script2.sh .
logger.txt prac_2 project project.tar.gz projScript.sh

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script2.sh ./prac_2/
script1.sh script2.sh script3.sh script4.sh script5.sh script6.sh

```

Рисунок 5 – Пример выполнения

Напишем сценарий, который с помощью цикла прочитает файл и выведет его содержимое.

```

prac_2 > $ script3.sh
1  #!/bin/bash
2
3  file="$1"
4
5  while IFS= read -r line; do
6      echo "$line"
7  done < "$file"
8
9

```

Рисунок 6 – Исходный код

```

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script3.sh logger.txt
Дата и время: Wed, Mar 19, 2025 5:05:42 PM
Список зарегистрированных пользователей:
Default
Public
semen
Время работы системы:
13516.96 seconds

```

Рисунок 7 – Пример выполнения

Напишем сценарий, который с помощью цикла выведет список файлов и директорий из текущего каталога, укажет, что есть файл, а что директория.

```
prac_2 > $ script4.sh
1  #!/bin/bash
2
3  for item in *; do
4      if [ -d "$item" ]; then
5          echo "$item – директория"
6      elif [ -f "$item" ]; then
7          echo "$item – файл"
8      fi
9  done
10
```

Рисунок 8 – Исходный код

```
LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script4.sh
  logger.txt – файл
  prac_2 – директория
  project – директория
  project.tar.gz – файл
  projScript.sh – файл
```

Рисунок 9 – Пример выполнения

Напишем сценарий, который подсчитает объем диска, занимаемого директорией. В качестве директории можно выбрать любую директорию в системе.

```
prac_2 > $ script5.sh
1  #!/bin/bash
2
3  du -sh $1
4
```

Рисунок 10 – Исходный код

```
LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script5.sh .
796M      .

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script5.sh prac_2
9.0K      prac_2
```

Рисунок 11 – Пример выполнения

Напишем сценарий, который выведет список всех исполняемых файлов в директории, для которых у текущего пользователя есть права на исполнение.

```
prac_2 > $ script6.sh
1  #!/bin/bash
2
3  find "$1" -maxdepth 1 -type f -executable -user "$(whoami)" -print
4
```

Рисунок 12 – Исходный код

```
LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script6.sh prac_2
prac_2/script1.sh
prac_2/script2.sh
prac_2/script3.sh
prac_2/script4.sh
prac_2/script5.sh
prac_2/script6.sh

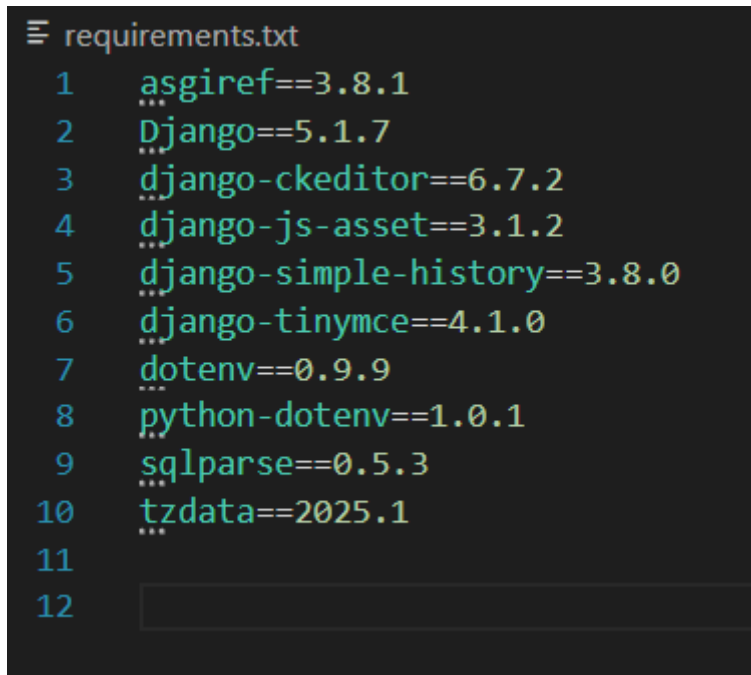
LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
⊗ $ ./prac_2/script6.sh
find: ‘’: No such file or directory

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./prac_2/script6.sh .
./projScript.sh
```

Рисунок 13 – Пример выполнения

Часть 2

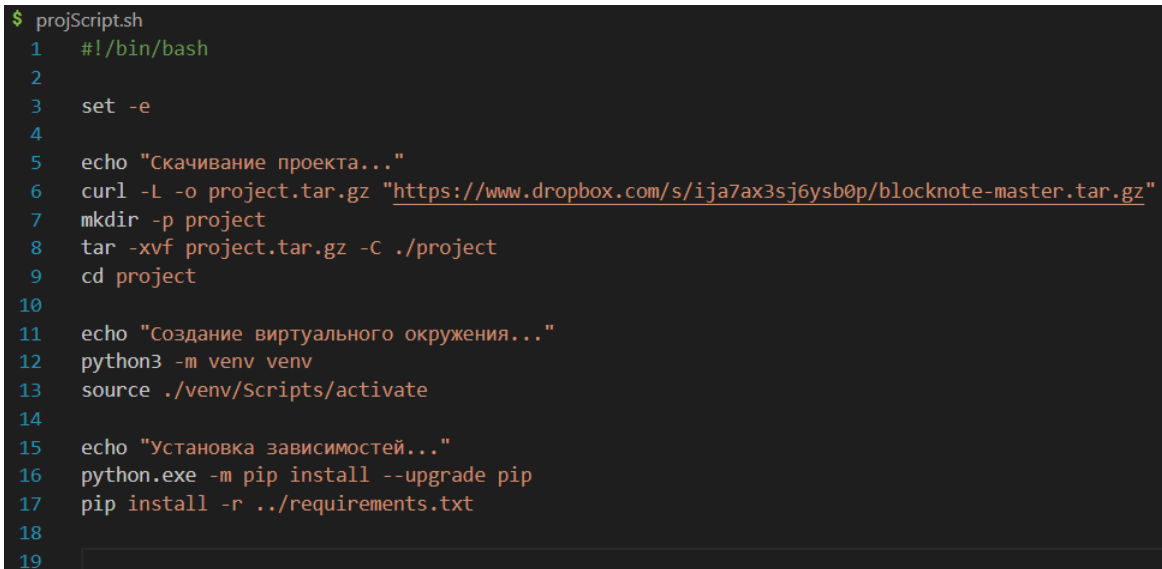
Определим зависимости проекта.

A screenshot of a code editor showing a file named 'requirements.txt'. The file contains 12 lines of dependencies, each preceded by a line number from 1 to 12. The dependencies are: asgiref==3.8.1, Django==5.1.7, django-ckeditor==6.7.2, django-js-asset==3.1.2, django-simple-history==3.8.0, django-tinymce==4.1.0, dotenv==0.9.9, python-dotenv==1.0.1, sqlparse==0.5.3, tzdata==2025.1, and an empty line at the end.

```
requirements.txt
1  asgiref==3.8.1
2  Django==5.1.7
3  django-ckeditor==6.7.2
4  django-js-asset==3.1.2
5  django-simple-history==3.8.0
6  django-tinymce==4.1.0
7  dotenv==0.9.9
8  python-dotenv==1.0.1
9  sqlparse==0.5.3
10 tzdata==2025.1
11
12
```

Рисунок 14 – Зависимости проекта

Напишем скрипт, который будет скачивать проект, создавать виртуальное окружение и настраивать его под проект.

A screenshot of a shell script named 'projScript.sh' in a code editor. The script contains 19 lines of code, numbered 1 to 19. The code performs the following steps: sets the shell to bash, sets error handling, echoes a message, downloads a tarball from a Dropbox link, extracts it into a 'project' directory, creates a virtual environment with python3, activates it, echoes another message, and finally installs the dependencies from the requirements.txt file using pip.

```
projScript.sh
1  #!/bin/bash
2
3  set -e
4
5  echo "Скачивание проекта..."
6  curl -L -o project.tar.gz "https://www.dropbox.com/s/ija7ax3sj6ysb0p/blocknote-master.tar.gz"
7  mkdir -p project
8  tar -xvf project.tar.gz -C ./project
9  cd project
10
11 echo "Создание виртуального окружения..."
12 python3 -m venv venv
13 source ./venv/Scripts/activate
14
15 echo "Установка зависимостей..."
16 python.exe -m pip install --upgrade pip
17 pip install -r ../requirements.txt
18
19
```

Рисунок 15 – Исходный код


```

LaHaine@ValeraMagistr MINGW64 ~/trpp_main (new_branch)
● $ ./projScript.sh
Скачивание проекта...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  131    100   131    0     0    208      0 --:--:-- --:--:-- --:--:--    208
100   17    100    17    0     0    13       0 0:00:01 0:00:01 --:--:--   2125
100  470     0   470    0     0   188      0 --:--:-- 0:00:02 --:--:--  52222
100 12.0M   100 12.0M    0     0 1147k      0 0:00:10 0:00:10 --:--:-- 1578k
blocknote-master/
blocknote-master/._DS_Store
blocknote-master/.DS_Store
blocknote-master/appengine/
blocknote-master/._.gitignore
blocknote-master/.gitignore
blocknote-master/static/
blocknote-master/._.gitattributes
blocknote-master/.gitattributes
blocknote-master/templates/
blocknote-master/._manage.py
blocknote-master/manage.py
blocknote-master/apps/
blocknote-master/apps/._DS_Store
blocknote-master/apps/.DS_Store
blocknote-master/apps/todoapp/
blocknote-master/apps/main/

```

Рисунок 16 – Пример части выполнения

Улучшим имеющийся скрипт, до скрипта, который может запускаться на новой системе.

```
$ projScript.sh
1  #!/bin/bash
2
3  set -e
4
5  echo "Установка Python и pip..."
6  sudo apt update
7  sudo apt install -y python3 python3-venv python3-pip
8
9  echo "Скачивание проекта..."
10 curl -L -o project.tar.gz "https://www.dropbox.com/s/ija7ax3sj6ysb0p/blocknote-master.tar.gz"
11 mkdir -p project
12 tar -xvf project.tar.gz -C ./project
13 cd project
14
15 echo "Создание виртуального окружения..."
16 python3 -m venv venv
17 source ./venv/Scripts/activate
18
19 echo "Установка зависимостей..."
20 python.exe -m pip install --upgrade pip
21 pip install -r ../requirements.txt
22
23 echo "Установка Django"
24 pip install django
25 django-admin startproject myproject
26 cd myproject
27
28 echo "Запуск проекта"
29 python manage.py makemigrations
30 python manage.py migrate
31 python manage.py runserver
32
33
```

Рисунок 17 – Исходный код

ВЫВОД

В ходе практической работы были изучены основы Bash-скриптов и их применение для автоматизации развертывания и запуска Django-проекта. Рассмотрены основные команды Bash, такие как работа с файлами и папками, управление переменными, проверка условий и использование циклов.

Практическое применение Bash-скриптов позволило автоматизировать процесс развертывания проекта и сделать запуск веб-приложения более удобным и быстрым.